Synchronization

1. A method for updating an asset in a source environment on a source node connected to a network, comprising the steps of:

receiving a synchronization call having a data argument and an asset type, the data argument defining changes to the asset, the asset being of the asset type;

selecting an adapter, associated with the asset type, based on the asset type;

passing the data argument to the adapter;

determining if the asset type is at least one of an entity data type and entity bean type;

determining a table, associated with the asset, based on the data argument;

retrieving at least one synchronization information object from a target environment on the target node;

creating a synchronization asset having a logic/data portion and an extended environment portion copied from the asset, the asset being updated with the synchronization information object;

establishing a connection between the target node and the source node, the source node being the origin of the asset; and

sending the synchronization asset over the network to the source node.

2. The process according to claim 1, wherein the synchronization call is generated external to the target node.

3. The process according to claim 1, wherein the synchronization call is generated internal to the target node.

4. The process according to claim 1, wherein the asset type is an entity data type and the argument data indicates that a synchronization should occur every time a database record in the table is changed.

5. The process according to claim 1, wherein the asset type is an entity data type and the argument data indicates that a synchronization should occur every time a specific set of database tables is changed.

6. The process according to claim 1, wherein the asset type is an entity data type and the argument data indicates that a synchronization should occur every time a database record in the table is changed and a certain user interface navigation has occurred.

7. The process according to claim 1, wherein the asset type is an entity data type and the argument data indicates that a synchronization should occur every time a database record in the table is changed and a certain connectivity event has occurred.

8. The process according to claim 1, wherein the asset type is an entity data type and the argument data indicates that a synchronization should occur every time a database record in the table is changed and a certain number of database record changes has occurred.

9. The process according to claim 1, wherein the asset type is an entity data type and the argument data indicates that a synchronization should occur every time a database record in the table is changed and a certain set of database changes have occurred resulting in a number of bytes being changed.

10. The process according to claim 1, wherein the asset type is an entity data type and the argument data indicates that a synchronization should occur every time a database record in the table is changed and at least one of a database record in the table is changed, a specific set of database tables is changed, a database record in the table is changed and a certain user interface navigation has occurred, a database record in the table is changed and a certain user interface navigation has occurred, a database record in the table is changed and a certain connectivity event has occurred, a database record in the table is changed and a certain number of database record changes has occurred, and a database record in the table is changed and a certain set of database changes have occurred that result in a number of bytes being changed.

11. The process according to claim 1, wherein the synchronization information object includes at least one of an insertion, a deletion, and updating of database records.

12. The process according to claim 1, wherein the connection is established through a component distribution server/asset distribution server.

13. The process according to claim 1, wherein the connection is established directly to the source node.

14. The process according to claim 1, wherein the connection is established indirectly to the source node through a chain of component distribution servers.

15. The process according to claim 15, wherein the source node includes at least one of a target node, an enterprise information system, a component distribution server/asset distribution server, and an edge server.

16. The process according to claim 15, wherein the connection is made using the machine address of the source node.

17. The process according to claim 1, wherein the source node includes an enterprise information system having at least one of an on-line retail/wholesale sales, marketing, and inventory system, an enterprise supply chain management systems, a product and/or content distribution systems, an on-line financial systems, a service providing system, an on-line human resource system, an on-line pay roll service, an on-line banking system, an airline reservation system, and a general business transaction system.

18. The process according to claim 1, wherein "said asset being updated" includes at least one of updating the extended environment portion, and updating the logic/data portion.

19. The process according to claim 1, wherein the connection is made through an enterprise information system deployment agent on the source node.

20. The process according to claim 1, wherein the synchronization information object includes at least one SQL query applicable to a source database table on the source.

21. The process according to claim 1, further including creating a "coalesced synchronization asset" from a first synchronization asset and a second synchronization asset.

22. The process according to claim 22, further including:

determining whether the first synchronization asset and the second synchronization conflict; and

rectifying the conflict using a compensating transaction.

23. A method for synchronizing an asset on a multi-tiered network, comprising:

receiving an asset synchronization request;

selecting a synchronization asset adapter associated with the asset;

retrieving synchronization information associated with the asset;

creating a synchronization asset; and

transferring the synchronization asset to a component distribution server.

24. The method of claim 23, wherein the asset is an entity bean type.

25. The method of claim 23, wherein the asset is an entity data type.

26. The method of claim 25, wherein the synchronization request is received every time at least one record in a database associated with the asset is changed.

27. The method of claim 26, wherein the synchronization information includes insertions, deletions, or modifications to the database associated with the asset.

28. The method of claim 27, wherein the synchronization information further comprises recorded SQL queries adaptively configured to be applied to a source database.

29. The method of claim 23, wherein the asset includes a logic/data part and an extended environment part, and the synchronization asset is created by updating the logic/data part and extended environment part of the asset.

30. A method for synchronizing an asset on a multi-tiered network, comprising:

receiving an asset synchronization request in a target environment;

determining a first asset type associated with the asset;

selecting a first synchronization asset adapter associated with the asset;

determining a table associated with the asset;

retrieving synchronization information associated with the asset;

creating a synchronization asset having the synchronization information;

transferring the synchronization asset to a component distribution server;

determining a source environment associated with the asset;

sending the synchronization asset to the source environment;

selecting a second synchronization asset adapter associated with the synchronization asset;

determining a second asset type associated with the synchronization asset; and

applying the synchronization information from the synchronization asset to the asset in the source environment.

31. The method of claim 30, wherein the asset is an entity bean type.

32. The method of claim 30, wherein the asset is an entity data type.

33. The method of claim 32, wherein the synchronization request is received every time at least one record in a database associated with the asset is changed.

34. The method of claim 33, wherein the synchronization information includes insertions, deletions, or modifications to the database associated with the asset.

35. The method of claim 34, wherein the synchronization information further comprises recorded SQL queries adaptively configured to be applied to a source environment database.

36. The method of claim 30, wherein the asset includes a logic/data part and an extended environment part, and the synchronization asset is created by updating the logic/data part and extended environment part of the asset.

37. The method of claim 30, further comprising:

transferring a plurality of synchronization assets to the component distribution server;

creating a coalesce synchronization asset from the plurality of synchronization assets; and

sending the coalesce synchronization asset to the source environment.

38. The method of claim 30, wherein the synchronization asset is sent to an enterprise information system deployment agent in the source environment.

39. The method of claim 30, wherein the synchronization request is sent to a client distribution agent in the target environment.

40. A system for synchronizing an asset on a multi-tiered network, comprising:

an asset synchronization request;

a target computer, including a new version of the asset, a first synchronization asset adapter, and a synchronization asset associated with the new version of the asset;

a component distribution computer, including a database associating assets with source computers; and

a source computer, including an old version of the asset and a second synchronization asset adapter.

41. The system of claim 40, wherein the target computer further includes a client distribution agent.

42. The system of claim 40, wherein the source computer further includes an enterprise information system deployment agent.

43. The system of claim 40, wherein the component distribution computer further includes a coalesce synchronization asset.

Target

1. A method of operating a computer system for targeting one or more digital assets on a distribution server connected to one or more networks so that the digital assets are compatible with one or more target nodes connected to the networks, the method comprising:

examining the one or more digital assets to determine an asset type of the digital asset;

if the asset type is Relational Data (RD), retrieving one or more where clauses of the digital asset;

executing a token replacement operation on the where clause to create a transformed where clause;

running a query on one or more tables specified in the digital asset using the transformed where clause, the query returning one or more returned records, the returned records correlating with the target node; and

storing the returned record in the digital asset.

2. The method of claim 1, wherein the target node is registered with the computer system.

3. The method of claim 1, wherein the target node is one of a class of target nodes registered with the computer system.

4. The method of claim 1, wherein the digital asset is in a neutral or normalized format.

5. The method of claim 1, wherein the where clause is stored in an extended environment section of the digital asset.

6. The method of claim 1, wherein the asset type is any one of the set of entity bean (EB), reference data (RD) and entity data (ED).

7. The method of claim 1, wherein the token replacement operation comprises a string replacement operation that retrieves a replacement value from a node specification corresponding to the targeted node.

8. The method of claim 1, wherein an SQL query using the transformed where clause query returns data that is placed in a logic/data (LD) section of the digital asset.

9. The method of claim 1, wherein the asset type is Relational Data (RD) and the where clause has tokens that need to be replaced using information in a node's registry of the target node.

10. The method of claim 1, wherein the asset type is Reference Data (RD) and the where clause has tokens that need to be replaced using information in a node's registry of the target node.

11. The method of claim 1, wherein an SQL query using the transformed where clause query returns data that is placed in a logic/data (LD) section of the asset.

12. The method of claim 1, wherein the asset type is Entity Bean (EB) and a query has to be performed for each target node in a set of target nodes.

13. The method of claim 1, wherein the asset type is Entity Data (ED) and a query has to be performed for each target node in a set of target nodes.

14. The method of claim 13, wherein the query queries data directly correlating to an individual end user.

15. The method of claim 13, wherein the query comprises the tokens, and wherein the target node provides filtering for the capture of client activity logging.

16. The method of claim 15, wherein the client activity logging data is recorded in the ED in order to synchronize such data back into the source environment.

17. The method of claim 13, wherein the query queries the status of the current orders for an individual customer.

18. A method for targeting a digital asset to a multi-tiered network node, comprising:

selecting a target asset adapter associated with the digital asset;

determining an asset type associated with the digital asset;

retrieving a descriptor from a first data structure associated with the digital asset;

transforming the descriptor using a token replacement operation having a token associated with the node;

running a query, using the transformed descriptor, on a table specified in the first data structure associated with the digital asset;

creating a second data structure using data returned by the query; and

inserting the second data structure into the first data structure associated with the digital asset.

19. The method of claim 18, wherein the digital asset is a Reference Data (RD) type.

20. The method of claim 18, wherein the digital asset is an Entity Data (ED) type.

21. The method of claim 18, wherein the first data structure comprises a Logic/Data (LD) part and an Extended Environment (EE) part.

22. The method of claim 21, wherein the descriptor is retrieved from the EE part.

23. The method of claim 21, wherein the second data structure is inserted into the LD part.

24. The method of claim 18, wherein the token replacement operation is a basic string replacement operation.

25. The method of claim 18, wherein the token is retrieved from a specification associated with the node.

26. The method of claim 18, wherein the token is retrieved from a registry associated with the node.

27. The method of claim 18, wherein the query is a SQL query and the data returned by the query are records.

28. The method of claim 18, wherein the table resides in a source tier, and the query is run in the source tier.

Adjust

1. A method for adjusting the distribution of an asset over a network, comprising the steps of:

receiving at least one metric;

determining a network optimization and at least one associated change requirement based on the metric and at least one model, the metric being input to the model;

changing a package specification to an changed package specification based on the change requirement, the changed package specification having at least one asset being packaged in at least one package;

implementing the change requirement; and

distributing the package over the network.

2. The method according to claim 1, wherein said receiving includes a synchronization process.

3. The method according to claim 1, wherein the metric includes at least one of: a data transfer between a plurality of agents, a transaction per second for computational environments, a condition of a secure network connection, a number of clients per second that have been accessing a distribution server, a number of router hops between nodes, a bandwidth provisioning cost of the links between nodes, and a physical distance of at least one asset from a target node

4. The method according to claim 1, wherein the model includes at least one of: a load balancing model, a QoS model, a security model, a performance model, a distribution model, a routing model, and a target node location model.

5. The method according to claim 1, wherein the change requirement includes at least one of: a re-location of at least one asset, a movement of at least one asset, a re-routing of at least one asset, a re-direction of at least one asset, a re-location of at least one computational request, a movement of at least one computational request, a re-routing of at least one computational request, a re-direction of at least one computation request, moving at least one asset to a cache, moving at least one asset closer to a target node, moving at least one asset to an alternate target environment, and a selection of an alternate computational environment in which to fulfill a request for execution.

6. The method according to claim 1, wherein the model is a load-balancing model and the metric includes at least one statistic associated with how a network node handles requests.

7. The method according to claim 1, wherein the model is a service level agreement model and the metric includes a "quality of service" token.

8. The method according to claim 1, wherein the model is a network routing optimization model and the metric includes at least one statistic associated with how the asset is distributed.

9. The method according to claim 1, wherein the change requirement includes changing the physical distance between at least one asset and a target network node.

10. The method according to claim 1, wherein the model is a network routing optimization model and the metric includes at least one statistic associated with how the asset is deployed.

Bridging

1. A method for maintaining communication between at least two executable system parts executing on at least two nodes of a multi-tiered network, comprising the steps of:

detecting a fault when at least one asset deployed on a local node attempts to access at least one resource on a remote node, the resource being accessed through an application programming interface;

determining whether the resource causing the fault is defined in a directory service that should be accessed on the remote node, and if so, providing a reference to the remote node;

if the resource is not defined in the directory service, determining whether the fault is a database interface fault; and

if the fault is not a database interface fault, determining whether the fault is at least one of a server fault and an object fault.

2. The method according to claim 1, wherein the reference includes a remote proxy object.

3. The method according to claim 1, wherein the reference includes at least one of a redirection, a server proxy, an object proxy, and an API proxy.

4. The method according to claim 1, wherein the reference includes a lookup mechanism, the lookup mechanism determining a source node associated with the asset.

5. The method according to claim 4, wherein the lookup mechanism includes a flat file, the flat file being edited when the asset is deployed to the local node to indicate that the asset requires a reference to a remote object.

6. The method according to claim 4, wherein the lookup mechanism is stored in a persistent database in the source node.

7. The method according to claim 1, wherein the fault is a database interface fault, and further including at least one of:

distributing at least one of an entity bean asset, an entity data asset, and a reference data asset; and

synchronizing at least one of an entity bean asset, an entity data asset, and a reference data asset.

8. The method according to claim 1, wherein the fault is a server fault, the server fault being associated with an attempt to reference an object on a server and the server is registered as requiring a proxy connection.

9. The method according to claim 8, further including connecting a proxy to a component distribution/asset distribution server that can proxy the request for the server.

10. The method according to claim 8, wherein the server resides on one of a component distribution/asset distribution server and a source node.

11. The method according to claim 1, wherein the application programming interface includes a J2EE API, the directory service includes a JDNI, and the database interface includes a JDBC.

12. The method according to claim 1, wherein the fault is an object fault, the object fault being associated with a request for an object that is a stub/proxy of an actual object.

13. The method according to claim 12, wherein the actual object resides in at least one of a source node and an intermediate target node.

14. The method according to claim 12, further including performing a proxy by at least one of a redirection and a proxy request.

15. The method according to claim 1, wherein the local node is at least one of a personal computer, a workstation, a pervasive device, a local server, a local area network server, a proxy server, an edge server, a general network servers, and an enterprise information system.

16. A method for maintaining communication between at least two executable system parts executing on at least two nodes of a multi-tiered network, the method comprising:

detecting a fault when at least one asset deployed on a local node attempts to access at least one resource on a remote node, the resource being accessed through an application programming interface; and

relaying the fault and an associated context between the two runnable system parts using a plurality of messaging schemes.

17. A method for bridging assets residing on multi-tiered network nodes, comprising:

receiving, from a first asset in a target environment, a request to access a second asset;

failing to access the second asset;

generating a fault;

creating a bridged computational environment associated with the second asset and an alternative environment; and

accessing the second asset in the alternative environment.

18. The method of claim 17, wherein the alternative environment is a source environment, a distribution tier environment, or an alternative target environment.

19. The method of claim 17, wherein the fault is a J2EE API fault.

20. The method of claim 19, wherein the second asset is a remote object defined in a JNDI.

21. The method of claim 20, wherein said creating includes providing a reference associated with the remote object to the first asset.

22. The method of claim 21, wherein said providing a reference is by redirection, server proxy, object proxy, or API proxy.

23. The method of claim 21, wherein said providing a reference is by lookup into a flat file having associations between assets and remote objects.

24. The method of claim 21, wherein said providing a reference is by lookup into a persistent database table having associations between assets and remote objects.

25. The method of claim 17, wherein the fault is a JDBC fault.

26. The method of claim 25, wherein the second asset is an enterprise bean, an entity data, or a reference data type.

27. The method of claim 26, wherein said creating includes distribution of the second asset.

28. The method of claim 26, wherein said creating includes synchronization of the second asset.

29. The method of claim 17, wherein the fault is a server fault associated with a server.

30. The method of claim 29, wherein the second asset is a server object.

31. The method of claim 30, wherein said creating includes connecting a proxy to the server.

32. The method of claim 31, wherein the proxy is a component distribution server.

33. The method of claim 31, wherein the proxy is in the source environment or the distribution environment.

34. The method of claim 31, wherein the bridged computational environment includes HTTP tunneling of IIOP traffic through a firewall associated with the server.

35. The method of claim 17, wherein the fault is an object fault.

36. The method of claim 35, wherein the second asset is a stub object associated with an actual object in the alternative environment.

37. The method of claim 36, wherein said creating includes connecting the stub object and the actual object.

38. The method of claim 17, wherein said accessing is transparent to the first asset.

39. A system for bridging assets residing on multi-tiered network nodes, comprising:

a first asset;

a second asset;

a target environment, including the first asset, an asset access adapter, and a fault handler;

an alternative environment, including the second asset; and

a bridged computational environment associated with the second asset and the alternative environment.

40. The system of claim 39, wherein the alternative environment is a source environment, a distribution tier environment, or an alternative target environment.

41. The system of claim 39, further comprising a server proxy, an object proxy, or an API proxy.

Streaming

1. A method for distributing changes to digital assets across a network, said method comprising:

determining an asset type of a first digital asset;

comparing the first digital asset to a prior digital asset to determine one or more deltas, the prior digital asset being a prior version of the first digital asset; the delta being a difference between the first digital asset and the prior digital asset;

evaluating the one or more of the deltas with one or more criteria to determine if the one or more delta assets should be created, the delta asset being a second digital asset containing the respective delta, the criteria determined by the asset type;

if the delta meets the criteria, creating the delta asset; and

marking the delta asset as a first delta asset of the first digital asset.

2. The method of claim 1, wherein the comparing is done at an Enterprise Information System (EIS) connected to the network.

3. The method of claim 2, wherein the EIS comprises one or more of the following: an on-line customer order entry system; an on-line retail/wholesale sales system, an on-line marketing system, an on-line inventory system, an enterprise supply chain management system, a product distribution system, a content distribution system, a television system, a phone system, an on-line financial system, a mortgage application system, an investing system, a stock trading system, a loan application system, a credit card account system, a service providing system, a medical service system, a legal service system, a real estate system, an engineering system, an education system, a distance leaning system, a technical

support system, an on-line human resource system, a pay roll services system, an on-line banking system, a banking system, a financial institution, a manufacturer, an airplane manufacturer, an internal corporate system, an airline reservation system; and a general business transacting system.

4. The method of claim 1, wherein the asset type comprises one of a static content (SC), presentational component (PC), transactional component (TC), and relational data (RD), with one or more data structures in the first digital asset, and wherein the comparing being done by a query of one or more of the data structures in the first digital asset and the prior digital asset to determine the delta, and the criteria is a threshold amount of data that has changed.

5. The method of claim 1, wherein the asset type comprises one of an entity bean (EB), and entity data (ED) and a reference data (RD), with one or more data structures in the first digital asset, and wherein the comparing being done by a query of one or more of the data structures in the first digital asset and the prior digital asset to determine the delta, and the criteria is a threshold amount of data that has changed.

6. The method of claim 1, wherein the first delta asset will be applied to the prior digital asset located on one or more target nodes connected to the network in order to create the first digital asset on the target node.

7. The method of claim 6, wherein one or more of the prior digital assets on the target is marked for removal after the respective first digital asset is created on the target node.

8. The method of claim 6, wherein one or more of the prior deltas on the target is marked for removal after the respective first digital asset is created on the target node.

9. The method of claim 1, wherein a set of two or more deltas is determined.

10. The method of claim 9, wherein the set is evaluated against a second criteria to determine whether the first delta asset is created.

11. The method of claim 10, wherein the second criteria prevents the first delta asset from being created if the aggregate of one or more of the deltas in the set are larger than the prior digital asset.

12. The method of claim 11, further comprising creating a new digital asset incorporating all of the deltas in the set.

13. The method of claim 1, further comprising coalescing two or more of the deltas into a coalesced delta that is used to create the first delta asset.

14. The method of claim 1, wherein said method is only executed on a node of the network if the first digital asset is registered with the node.

15. The method of claim 1, wherein one or more of the deltas has information used to synchronize a source node with changes at one or more target nodes, the source and target nodes connected to the network.

16. The method of claim 1, wherein one or more of the deltas is stored along with the prior digital asset in order to provide the deltas for some nodes and the deltas combined with the prior digital asset for those nodes that do not have the prior asset.

17. The method of claim 16, wherein the deltas are applied to the prior digital asset for distribution to nodes that do not contain the prior digital asset.

EEP

1. A package structure distributed over a communications network, comprising:

at least one representation of an asset, the asset having a logic/data portion and an asset extended environment portion; and

a package extended environment having package information associated with at least one asset.

2. The package structure according to claim 1, wherein the representation is an identifier of the asset.

3. The package structure according to claim 1, wherein the representation is the asset.

4. The package structure according to claim 1, wherein the package information includes common information associated with all of the assets having representations in the package structure.

5. The package structure according to claim 1, wherein the package information includes at least one descriptor, the descriptor describing the assets having representations in the package structure.

6. The package structure according to claim 1, wherein the package information includes an extended markup language (XML) file.

7. The package structure according to claim 1, wherein the package information includes at least one common descriptor.

8. The package structure according to claim 7, wherein the common descriptor includes at least one of a package name, a package address, a package size, a volatility, a runnablilty, a user type, a version, a package level security descriptor, and a pricing level descriptor.

9. The package structure according to claim 1, wherein the package information includes at least one package dependency descriptor, the package dependency descriptor indicating other information on which the package structure depends.

10. The package structure according to claim 1, wherein the package information includes at least one package level reference descriptor.

11. The package structure according to claim 10, wherein the package level reference descriptor includes a package level reference link descriptor.

12. The package structure according to claim 11, wherein the package level reference link descriptor includes a world-wide-web address having contents used for processing of the package structure.

13. The package structure according to claim 11, wherein the package level reference link descriptor includes a world-wide-web address having contents used during execution of all the digital assets having representations in the package structure.

14. The package structure according to claim 10, wherein the package level reference descriptor includes a package level reference file descriptor.

15. The package structure according to claim 14, wherein the package level reference file descriptor includes a unique fully qualified name of a file required for reference by the package.

16. The package structure according to claim 10, wherein the package level reference descriptor includes a package level reference directory descriptor.

17. The package structure according to claim 16, wherein the package level reference directory descriptor includes additional address information used to locate at least one asset having a representation in the data structure.

18. The package structure according to claim 17, wherein the additional address information includes at least one of a root level directory name, a parent level directory name, a directory structure, a leaf directory level identification, and a director path.

19. The package structure according to claim 1, wherein the package information includes at least one pricing descriptor.

20. The package structure according to claim 19, wherein the pricing descriptor includes pricing information associated with at least one of a price, a price scheme, a subscription price scheme, a pay to own price scheme, a pay to use price scheme, a one time payment price scheme, a payment detail, and payment method.

21. The package structure according to claim 1, wherein the package information includes at least one security descriptor.

22. The package structure according to claim 21, wherein the security descriptor includes a package level description of at least one of an encryption function, an authorization function, and an access control function.

23. A method for creating a package structure, comprising the steps of:

storing at least one representation of an asset in the package structure, the asset having a logic/data portion and an asset extended environment portion; and

storing package information associated with at least one asset in a package information section of the package structure.

24. The method according to claim 23, further comprising the step of transmitting the package structure over at least one network connection.

25. The method according to claim 23, further comprising the step of receiving the package structure from at least one network connection.

26. The method according to claim 23, further comprising the step of saving the package structure in at least one system memory.

27. The method according to claim 26, wherein the system memory resides on at least one enterprise information system.

28. The method according to claim 27, wherein the enterprise information system includes at least one of an on-line customer order entry system; an on-line retail/wholesale sales system, an on-line marketing system, an on-line inventory system, an enterprise supply chain management system, a product distribution system, a content distribution system, a television system, a phone system, an on-line financial system, a mortgage application system, an investing system, a stock trading system, a loan application system, a credit card account system, a service providing system, a medical service system, a legal service system, a real estate system, an engineering system, an education system, a distance leaning system, a technical support system, an on-line human resource system, a pay roll services system, an on-line banking system, a banking system, a financial institution, a manufacturer, an airplane manufacturer, an internal corporate system, an airline reservation system; and a general business transacting system.

29. A system for creating a package structure, comprising:

a memory means;

a means for storing at least one representation of an asset in the package structure stored on the memory means, the asset having a logic/data portion and an asset extended environment portion; and

a means for storing package information associated with at least one asset in a package information section of the package structure stored on the memory means.

30. A medium for storing instructions adapted to be executed by a processor to perform the steps of:

storing at least one representation of an asset in the package structure, the asset having a logic/data portion and an asset extended environment portion; and

storing package information associated with at least one asset in a package information section of the package structure.

1. EXTENDED ENVIRONMENT DATA STRUCTURE FOR DISTRIBUTED DIGITAL ASSETS OVER A MULTI-TIERS COMPUTER NETWORK

2. SERVER SYSTEM AND METHOD FOR DISCOVERING DIGITAL ASSETS IN ENTERPRISE INFORMATION SYSTEMS

3. SERVER SYSTEM AND METHOD FOR EXPORTING DIGITAL ASSETS IN ENTERPRISE INFORMATION SYSTEMS

4. SERVER SYSTEM AND METHOD FOR DISTRIBUTING AND SCHEDULING MODULES TO BE EXECUTED ON DIFFERENT TIERS OF A NETWORK

5. DATA STRUCTURE, ARCHITECTURE APPARATUS, AND PROGRAM PRODUCT CAPABLE OF BEING DISTRIBUTED TO AND EXECUTED ON DIFFERENT NETWORK DEVICES AND ON VARIOUS COMPUTER PLATFORMS AND ENVIRONMENTS

6. SYSTEM AND METHOD FOR TRANSACTIONAL DEPLOYMENT J2EE WEB COMPONENTS, ENTERPRISE JAVA BEAN COMPONENTS, AND APPLICATION DATA OVER MULTI-TIERED COMPUTER NETWORKS

7. SYSTEM AND METHOD FOR DISTRIBUTING ASSETS TO MULTI-TIERED NETWORK NODES

8. METHOD AND SYSTEM FOR DEPLOYING AN ASSET OVER A MULTI-TIERED NETWORK

9. SYSTEM AND METHOD FOR TRANSLATING AN ASSET FOR DISTRIBUTION OVER MULTI-TIERED NETWORKS

10. SYSTEM AND METHOD FOR SYNCHRONIZING ASSETS ON MULTI-TIERED NETWORKS

11. METHOD AND SYSTEM FOR DEPLOYING